EMSCB:
European Multilaterally Secure Computing Base

DemoCD User Guide

for Turaya-Crypt / Turaya-VPN

June 19, 2006

Contents

1 Overview **3**
- 1.1 EMSCB . . . 3
- 1.2 Turaya-VPN . . . 3
- 1.3 Turaya-Crypt . . . 4

2 System Requirements **6**

3 Starting the CD **7**

4 Demo Applications **8**
- 4.1 Turaya-VPN . . . 9
- 4.2 Turaya-Crypt . . . 10
  - 4.2.1 Pre-configured Encrypted Device . . . 10
  - 4.2.2 Creating New Encrypted Devices . . . 12
- 4.3 TPM Server / Client . . . 14

A Known Issues and Problems **16**

Chapter 1

Overview

1.1 EMSCB

European Multilaterally Secure Computing Base (EMSCB) aims at developing a trustworthy computing platform with open standards that solves many security problems of conventional platforms. The platform deploys

- hardware functionalities provided by Trusted Computing,
- a security kernel based on a microkernel and
- an efficient migration of existing operating systems.

In the sense of multilateral security, the EMSCB platform allows the enforcement of security policies of different parties, i.e., end-users as well as industry. Consequently, the platform enables the realization of various innovative business models, while averting the potential risks of Trusted Computing platforms concerning privacy issues. The source code of the EMSCB platform will be published under an open-source license.

The DemoCD at hand documents the current development status of the components Turaya-VPN and Turaya-Crypt of the EMSCB project.

You can find the EMSCB webpage at http://www.emscb.org.

1.2 Turaya-VPN

Turaya-VPN is a Virtual Private Network client which is running in parallel to, but completely isolated from, a legacy operating system, i.e., Linux in this case. Certificates and keys as well as cryptographic operations are thus protected from hostile access by Linux and any malicious software such as worms and viruses.

Turaya-VPN is based on EMSCB technology and is interoperable with standard VPN servers (IPsec) and fully transparent to the user. It integrates a firewalling component and network configuration via the Dynamic Host Configuration Protocol (DHCP).

We have set up a demonstration webserver which is part of a demonstration intranet accessible through the Turaya-VPN client. The network communication to this webserver is encrypted with a secret key residing in the VPN component.

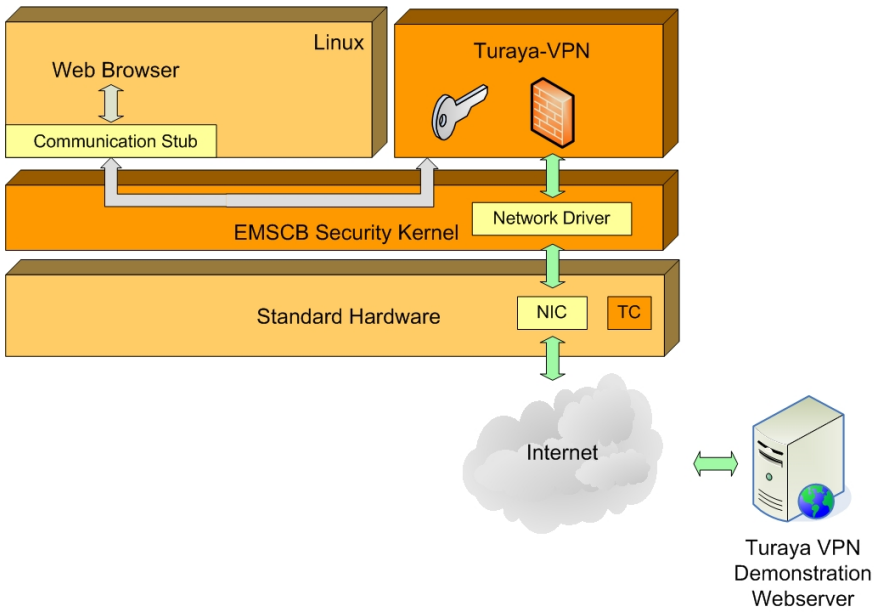

Figure 1.1: Architecture of Turaya-VPN

Future developments of the Turaya-VPN client will include Trusted Computing support to additionally "bind" the keys and certificates to a specific platform configuration.

For further information visit the EMSCB webpage at http://www.emscb.org.

1.3 Turaya-Crypt

Turaya-Crypt provides a secure device encryption for Linux. In contrast to the conventional device encryption mechanism provided by the Linux kernel, Turaya-Crypt is based on EMSCB technology to strongly isolate security-critical key information and cryptographic operations from the Linux operating system to prevent unauthorized access.

Turaya-Crypt handles security-critical information (e.g., keys) and cryptographic operations used by a wrapper module of the Linux device encryption mechanism. It acts as a decryption/encryption service running in parallel to the Linux operating system. Before the service can be used within Linux, the user has to provide a passphrase to Turaya-Crypt using a Trusted GUI, which cannot be accessed or manipulated from Linux. The passphrase is used to derive the encryption key that is used to encrypt and decrypt data.

Turaya-Crypt prevents any Linux application from accessing security-critical information and operations. Therefore, it clearly enhances the security of previous hard-disk encryp-

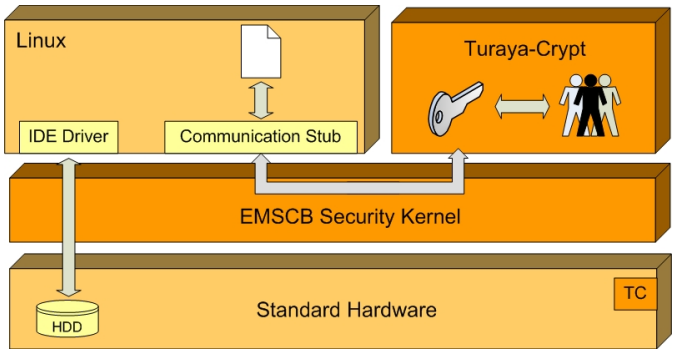

Figure 1.2: Architecture of Turaya-Crypt

tion schemes and provides full usability while implementing the standard Linux encryption interface.

Because of this isolation and protection mechanism, the user interface of Turaya-Crypt is split into two parts. One resides within Linux and can be used to create encrypted devices or virtual encrypted file systems. The other interface is outside of Linux and is used to provide the passphrase. These two types of user interfaces are strongly isolated as well, thus no program running within Linux can neither access nor manipulate the user interface used to enter the passphrase.

For further information visit the EMSCB webpage at http://www.emscb.org.

Chapter 2

System Requirements

- 32-bit Intel Pentium processor or better (IA32 586 compatible)
- 512 MB RAM at minimum
- CD-ROM drive from which the system can be booted
- VESA 2.0 or higher compatible graphics card
- PS/2 or USB mouse

Note: This DemoCD is a snapshot of work in progress and not a finished product. It has been tested on a limited amount of different configurations only and cannot be guaranteed to run on all systems that meet the mentioned requirements.

Chapter 3

Starting the CD

Insert the Turaya DemoCD into the CD-ROM drive of your computer and boot from CD. (You might need to change the boot order of your drives in the BIOS. Consult your computer manual in case you do not know how to do this.)

The bootmenu will ask you to select a resolution for the graphical window system of the Turaya Demo. Choose the graphics resolution which fits best to your system. 1024x768 should work in most cases.

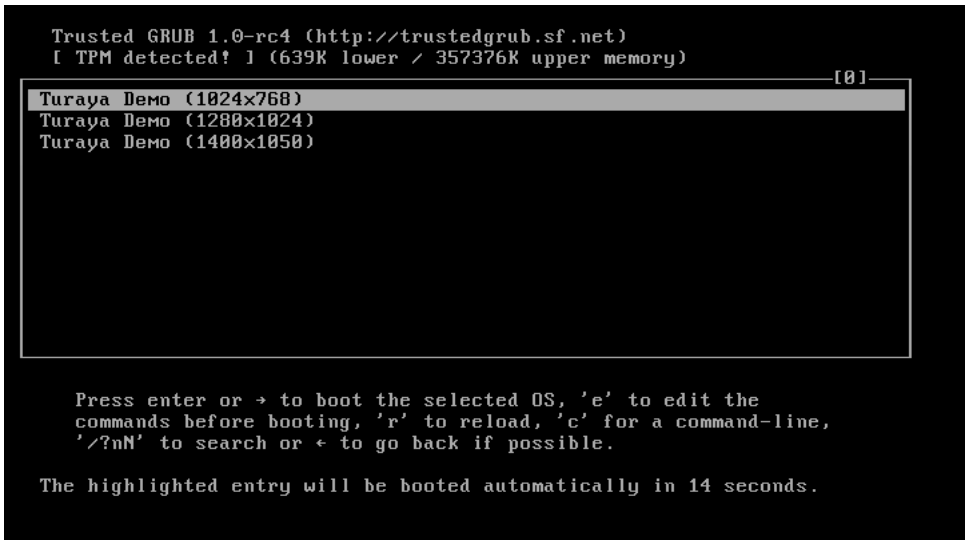
Trusted GRUB 1.0-rc4 (http://trustedgrub.sf.net)
[TPM detected!] (639K lower / 357376K upper memory)
[0]
Turaya Demo (1024x768)
Turaya Demo (1280x1024)
Turaya Demo (1400x1050)
Press enter or → to boot the selected OS, 'e' to edit the
commands before booting, 'r' to reload, 'c' for a command-line,
'/?nN' to search or ← to go back if possible.
The highlighted entry will be booted automatically in 14 seconds.

Figure 3.1: Screenshot of the bootmenu.

We use Trusted GRUB[1] as bootloader. If a Trusted Platform Module (TPM) is detected, it will be used to measure the loaded components. The measurement result will be stored within the TPM and can later be displayed in the TPM Client. Note that this is only a demonstration and no information on this CD is actually bound to a specific TPM. Thus, you can use the DemoCD on any PC, TPM-enabled or not.

[1] http://trustedgrub.sourceforge.net

Chapter 4

Demo Applications

After a few moments of loading, you will see two windows on the screen, i.e., a TPM Client[1] and a Linux window. The whole graphical display is controlled by the Trusted GUI, i.e., the window system server on the current DemoCD. You will find the Linux system running completely in its own window; it may take some time to load.

Figure 4.1: Screenshot after booting the EMSCB demo.

When the Linux desktop appears, click in the Linux window to have it gain the input focus. To escape from the Linux window again, you have to press the `[Pause]` key on your keyboard. This will change the mouse pointer from the Linux window to the overall window system

[1]The TPM Client will not appear if your system does not have a TPM or if the provided TPM is not supported.

again. Note that every time you click in the Linux window, the mouse pointer will be caught and you have to press the [Pause] key to leave the Linux window in order to click on the other windows outside of Linux.

4.1 Turaya-VPN

Click on the icon called "Turaya-VPN Demo" (see Figure 4.2).

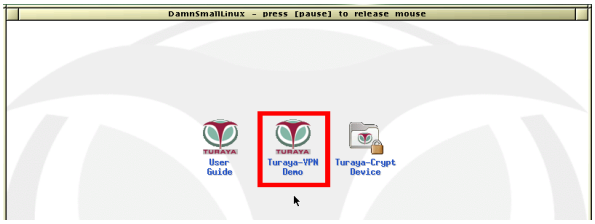

Figure 4.2: Start Turaya-VPN demo.

The Firefox browser will open and show a local welcome page (see Figure 4.3). Follow the link at the bottom to connect to the Turaya-VPN demonstration webserver. A connection to the Turaya Demonstration Virtual Private Network (VPN) will then be established.[2]

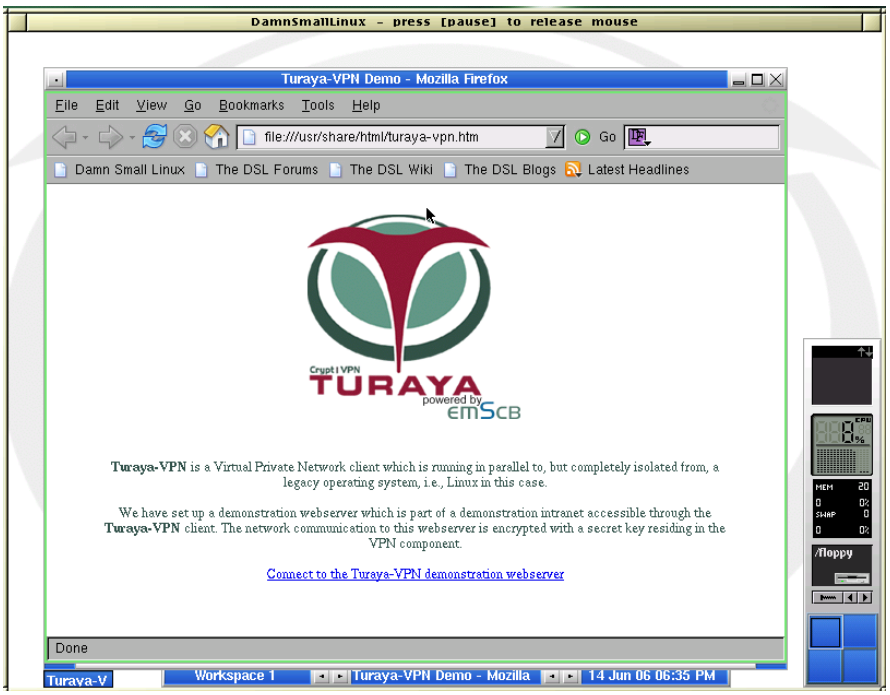
DamnSmallLinux - press [pause] to release mouse
Turaya-VPN Demo - Mozilla Firefox
File Edit View Go Bookmarks Tools Help
file:///usr/share/html/turaya-vpn.htm
Go
Damn Small Linux
The DSL Forums
The DSL Wiki
The DSL Blogs
Latest Headlines
Crypt! VPN
TURAYA
powered by emSCB
Turaya-VPN is a Virtual Private Network client which is running in parallel to, but completely isolated from, a legacy operating system, i.e., Linux in this case.
We have set up a demonstration webserver which is part of a demonstration intranet accessible through the Turaya-VPN client. The network communication to this webserver is encrypted with a secret key residing in the VPN component.
Connect to the Turaya-VPN demonstration webserver
Done
Turaya-V
Workspace 1
Turaya-VPN Demo - Mozilla
14 Jun 06 06:35 PM
/floppy

Figure 4.3: Welcome page of the Turaya-VPN demo.

[2]Please note that it may take up to 1 minute until the DHCP-client integrated in the Turaya-VPN has gathered the necessary network information to access the internet.

Technically speaking, the HTTP request to the private IP address 10.9.8.7 is routed through the EMSCB Security Kernel to the Turaya-VPN component. The Turaya-VPN component then identifies the IP address 10.9.8.7 as the destination address to the Demonstration VPN and encrypts the request. The responses back from the Turaya Demonstration VPN will be transparently decrypted and again handed over to the browser.

4.2 Turaya-Crypt

Let's take a look at the functions of Turaya-Crypt. On this DemoCD, Turaya-Crypt is executed in single-user single-key mode, i.e., there is only one user who can use Turaya-Crypt at the same time using a single key for all decryption and encryption operations. The user has to provide a passphrase, from which the key will be derived. The DemoCD already contains a pre-configured encrypted virtual device. We describe its usage in the following section.

4.2.1 Pre-configured Encrypted Device

This DemoCD contains a pre-configured encrypted virtual device, which can be used directly when running the DemoCD. Thus, there is no need to modify any hard-drive partition on your system. On the Linux desktop, you can find an icon called "Turaya-Crypt Device". When first started, this icon will display a lock symbol, which indicates that the device is encrypted and cannot be accessed (see Figure 4.4). If you click on the icon, the Firefox browser will start and open the corresponding directory. Since the device is not mounted yet, no containing data will be displayed.

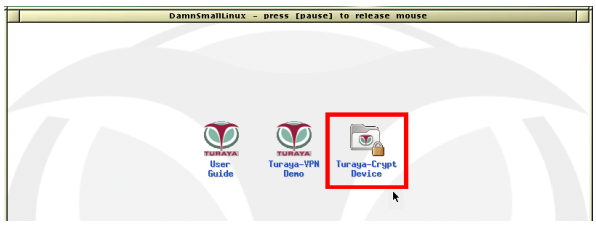

Figure 4.4: At the beginning, the Turaya-Crypt Device is locked.

To access the encrypted device, right-click on the icon and select the mount option:

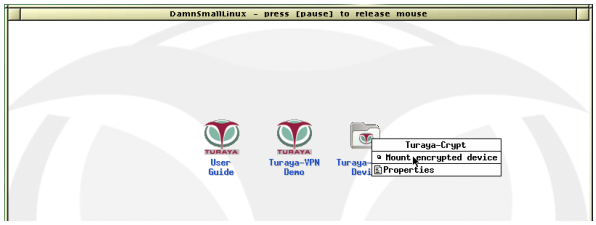

Figure 4.5: Mounting the Turaya-Crypt Device.

Now, Turaya-Crypt will ask you for the passphrase. Press `[Pause]` to escape from the Linux window and click on the passphrase dialog of Turaya-Crypt. Now, enter the passphrase (you have to enter the passphrase twice for confirmation purpose).

Figure 4.6: Passphrase dialog of Turaya-Crypt.

Passphrase: **emscb**

If you entered the passphrase correctly, Turaya-Crypt will mount the device and the lock symbol will disappear:

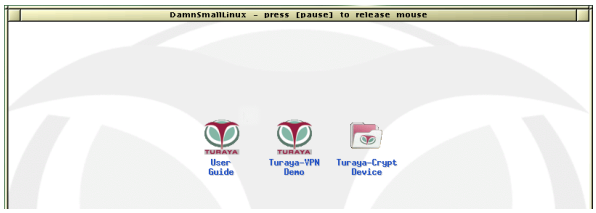

Figure 4.7: Turaya-Crypt Device has been mounted.

Now, all data read from the device will be transparently decrypted, while all data written to the device will be transparently encrypted, respectively. Left-click on the icon and the Firefox browser shows you the content of the encrypted device. You should see some PDF documents stored in the device (similar to Figure 4.8). Click on a document link and a PDF viewer will start and display the document.

You can unmount the device again by right-clicking the icon and choosing the unmount option. The lock symbol will then appear and all data contained in the device will remain encrypted until you mount the device again.

The pre-configured device demonstrates the usage of Turaya-Crypt for a simple case. In the following, we describe how to create new encrypted devices and mount or unmount them manually. However, this is considered for advanced and more experienced users only.

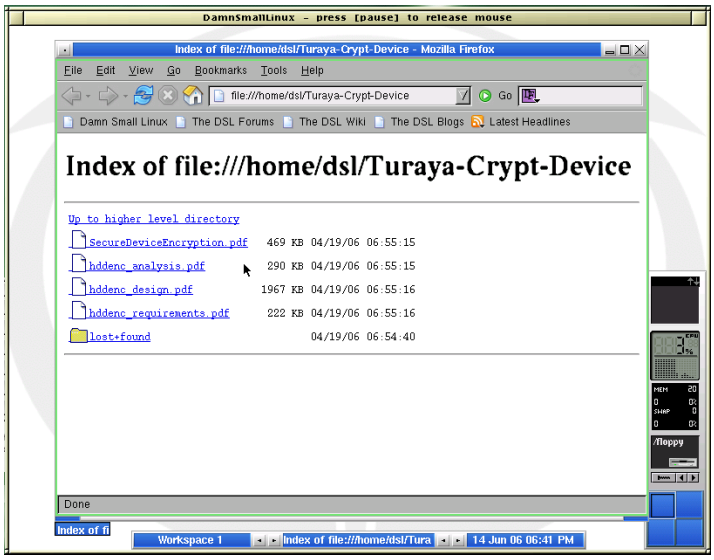
DamnSmallLinux - press [pause] to release mouse
Index of file:///home/dsl/Turaya-Crypt-Device - Mozilla Firefox
File Edit View Go Bookmarks Tools Help
file:///home/dsl/Turaya-Crypt-Device
Damn Small Linux The DSL Forums The DSL Wiki The DSL Blogs Latest Headlines
Index of file:///home/dsl/Turaya-Crypt-Device
Up to higher level directory
SecureDeviceEncryption.pdf 469 KB 04/19/06 06:55:15
hddenc_analysis.pdf 290 KB 04/19/06 06:55:15
hddenc_design.pdf 1967 KB 04/19/06 06:55:16
hddenc_requirements.pdf 222 KB 04/19/06 06:55:16
lost+found 04/19/06 06:54:40
Done
Workspace 1
14 Jun 06 06:41 PM

Figure 4.8: Contents of the mounted Turaya-Crypt Device.

4.2.2 Creating New Encrypted Devices

Besides using the pre-configured encrypted device, you can create new encrypted devices. However, we recommend to create only virtual devices, since this will not result in any modification of your system[3]. You can use the "`turaya-crypt`" Linux command to create, mount and unmount encrypted devices manually. To enter the Linux command line mode, click in the DamnSmallLinux window and start a shell by clicking on the "ATerminal" icon. When you create or mount an encrypted image file, the passphrase dialog will appear, unless you have provided the passphrase at a previous mount operation. Turaya-Crypt will "forget" the passphrase each time you unmount all encrypted devices. Thus, if you are going to mount a device again after you have unmounted all devices, you will have to re-enter the passphrase. Remember to use the same passphrase as you have chosen the first time, otherwise the decryption will fail because of a different key derivation.

Execute "`turaya-crypt`" for information about the usage of the Turaya-Crypt device encryption. "`turaya-crypt -h`" brings up a more detailed list:

```
usage: /bin/turaya-crypt <command>
  available commands:
      (-c | --create) destination [size]
         destination
            filename of the encrypted disc or device
            e.g. temp.img or /dev/hda2
         size
            required if destination is a filename
```

[3]Note that if you create an encrypted partition on your hard-drive, data previously stored in this partition will be overwritten. You should make a backup in order to prevent any data loss.

```
      size of the resulting image in megabytes, minimum: 2
(-m | --mount) destination mountpoint
   destination
      filename of the encrypted disc or device
   mountpoint
      the directory where the image file should be mounted
(-u | --umount) mountpoint
   mountpoint
      the directory where the image file is mounted
(-h | --help)
(-v | --version)
--debug
   enable debug mode
```

Let's give an example. You might have some files that you want to store in a secure way.

At first, create a Turaya-Crypt *image file*. This is the space where your data will be stored. Let's assume that 2 megabytes of storage will be enough for your demands. Execute the command

```
turaya-crypt -c /home/dsl/storage.img 2
```

where "`-c`" is the short form of "create", "`storage.img`" is the name of the image file (that you can choose arbitrarily and that will be stored in your home directory "`/home/dsl`"), and `2` is the size of the image file in megabytes.

Before the image file can be used to securely store data, it has to be *mounted*. This means that you have to specify a path in the file system that you want to use for direct access to the image file.

For this, create a new directory using the command

```
mkdir /home/dsl/secret
```

This directory will be the *mountpoint* of the image file – that is, the path that can be used for direct access to the image file as soon as it is mounted with the command

```
turaya-crypt -m /home/dsl/storage.img /home/dsl/secret
```

where "`-m`" is the short form of "mount", "`/home/dsl/storage.img`" is the location of the image file, and "`/home/dsl/secret`" is the new path for direct access to the secret storage ("the mountpoint of the image file").

Now you are ready to go! You can simply save your files that need to be stored securely to "`/home/dsl/secret`". This directory can now be used just like any other directory, with the difference that all data in "`/home/dsl/secret`" will be automatically encrypted (when

writing files) and decrypted (when reading files) by the Turaya-Crypt service.

When you are done, you can "unmount" the image file so that the path for direct access to the image file is no longer available (which also means that your secret data in the image file can no longer be accessed!). This is achieved by executing

```
turaya-crypt -u /home/dsl/secret
```

where "`-u`" is the short form of "unmount" and "`/home/dsl/secret`" is the mountpoint of the image file.

If you want to access your data (or store new data) in the image file again at a later time, you just have to mount the image file by executing "`turaya-crypt -m /home/dsl/storage.img /home/dsl/secret`" again and find your files in the directory "`/home/dsl/secret`". Of course, this would only be possible if you had logged in at the beginning with the same password that you used when you created the image file – no other user will be able to access the encrypted data.

Note: Since this is a DemoCD with a live starting system that uses a ramdisk as filesystem, all files that you will save within the DemoCD filesystem will be lost when you reboot the system. To keep your files, i.e., "storage.img", copy the file to a different computer system (e.g. using the Linux ssh command) or mount an existing physical hard disk partition where you can copy your image file. However, this is left to the more experienced user and will not be described in detail here.

4.3 TPM Server / Client

The DemoCD also features a TPM server, which supports three different TPM vendors: Infineon TPM 1.1b and 1.2, National Semiconductor (NSC/NSM) TPM 1.1b (present in IBM / Lenovo Thinkpads T43) and Atmel TPM 1.1b (present in IBM / Lenovo Thinkpads T41p).

The server is booted automatically during the DemoCD boot process and looks for either one of the TPMs. If it has found a TPM, it activates itself and listens for TPM commands (otherwise it quits immediately).

In order to use the TPM server in our secure environment, a small demonstration TPM client is included as well.[4] If the TPM server has found a TPM, the TPM client looks for the server and provides five TPM commands as a simple example:

- **Read PCR values**: Lists the Platform Configuration Registers (PCR) and their content.
- **Read public key**: Queries the TPM for its public key part.
- **Read version**: Displays the TPM version.
- **Get TPM random numbers**: Requests some random numbers from the hardware random number generator of the TPM.

[4] The TPM client will only be started if a supported TPM chip can be found on the computer system.

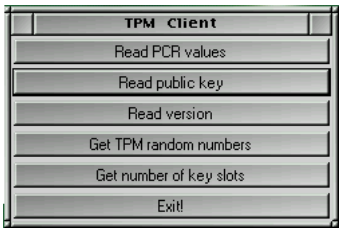

Figure 4.9: TPM client application

- **Get number of key slots**: Queries the TPM for its amount of key slots.
- **Exit!**: Quits the TPM client. The TPM server itself will keep running.

Appendix A

Known Issues and Problems

This DemoCD is a snapshot of work in progress and not a finished product. Known problems include:

- When clicking on the DamnSmallLinux window (in the window system mode of the DemoCD) for the first time, the mouse pointer may not react for 1-2 seconds.
- USB mouse on IBM ThinkPad may not work properly unless you touch the TrackPoint or the TouchPad once.
- Currently, the window system supports the German keyboard layout only.